NASA Technical Memorandum 107702

ON NEURAL NETWORKS IN IDENTIFICATION AND CONTROL OF DYNAMIC SYSTEMS

Minh Phan
Jer-Nan Juang
David C. Hyland

June 1993

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-0001

On Neural Networks in Identification and Control of Dynamic Systems

Minh Phan [1]
Lockheed Engineering & Sciences Co., Hampton, Virginia

Jer-Nan Juang [2]
NASA Langley Research Center, Hampton, Virginia

and

David C. Hyland [3]
Harris Corporation, Melbourne, Florida

The paper presents a discussion on the applicability of neural networks in the identification and control of dynamic systems. Emphasis is placed on the understanding of how the neural networks handle linear systems and how the new approach is related to conventional system identification and control methods. Extensions of the approach to non-linear systems are then made. The paper explains the fundamental concepts of neural networks in their simplest terms. Among the topics discussed are feedforward and recurrent networks in relation to the standard state-space and observer models, linear and non-linear auto-regressive models, linear predictors, one-step ahead control, and model reference adaptive control for linear and non-linear systems. Numerical examples are presented to illustrate the application of these important concepts.

1. Introduction

System identification and control are two related fields that have received considerable development in the last few decades. System identification deals with the problem of finding a mathematical description of a physical system from experimental data. Control theory devises ways to influence the system in a desirable and predictable manner. Typical control objectives are pointing control, vibration suppression, and tracking control. System identification provides the necessary mathematical model of a system for a particular control scheme to be designed. In turn, information gathered during the control process can be used to evaluate the validity of the assumed model. Existing system identification and control methods are based on mathematical systems theory, which first deals with deterministic then stochastic systems. For the most part, the systems under study are idealized. They are linear, time-invariant, and often assumed to be noise-free. When noises are present, they are assumed to be white, zero-mean, and with known characteristics. These assumptions are often justified because less idealized assumptions tend to render the analysis mathematically intractable.

In practice, however, all systems are affected by noises and non-linearities which may lead to instabilities for control laws that are based on idealized models. This motivates the development of a variety of control approaches which, if classified according to the amount of information required to design a controller, can be broadly divided in three classes. They are model-independent control, robust control, and adaptive control. A model-independent controller seeks to guarantee stability of the closed-loop system independently of the system model. As implied, such a design does not require that the system be known in advance. Robust controllers can tolerate certain specific variations about some nominal model, which is required to be known with somewhat accuracy, and variations about the nominal model need to be quantified. While a nominal model may be obtained analytically or experimentally, meaningful characterization of the variations about the nominal model is often difficult to obtain. In both model-independent control and robust control, there is a trade-off between stability robustness and performance. This is because performance requires that the system is known with certainty. If for some reason such knowledge is in error, then the designed optimal performance will not be achieved and instabilities may occur. Striking somewhat of a balance between the two control approaches is adaptive control which involves some level of on-line parameter estimation, where knowledge of the system being controlled is gained during the control process. The estimated parameters can be either the system parameters or the controller gains. In the former case, known as indirect adaptive control, the parameters representing a mathematical model of the system are identified on-line, and the control input is then computed. In the latter case, known as direct adaptive control, the system identification step is bypassed and the controller gains are directly updated at each time step. Adaptive control identifies the appropriate parameters of the system only for the purpose of control, thus offers a meaningful way to integrate system identification and control in one package. Adaptive control also offers the potential ability to handle systems with changing dynamics by constantly identifying and adjusting the control action accordingly.

Recently, there has been a substantial amount of interest in the field of neural networks. As a collection of interconnected neurons, a multi-layer neural network with appropriate weights has been shown to be able to approximate any input-output function. Consequently, the neural network is a natural candidate in the area of identification and control of both linear and non-linear systems.[1-5] The neural networks are typically implemented in the adaptive form, and thus possess similar attributes of adaptive control. The main objective of this paper is to examine the implication of the neural networks approach for linear systems, and to see how this approach is similar to and different from conventional methods. Only after a firm grasp of how the neural networks treat the linear problem, extensions to the non-linear problem can then be made, and the potential benefits of the non-linear approach can be better understood and appreciated. The extent to which linear approaches can handle non-linear systems can also be revealed. To this end, basic concepts in neural network will be presented and whenever possible, direct connection to existing system identification and control methods are made. Linear system identification techniques and adaptive control theory will be heavily drawn upon to make this connection.[6,7] In this paper, we focus on the role of the neural networks as applicable to structural system identification and control as opposed to other fields such as pattern recognition, image processing, and computer science. The recently developed Observer/Kalman filter identification (OKID) algorithm will also be discussed in the context of the neural networks.[8-10] Potential applicability of the existing techniques to non-linear problems will be examined. Several numerical examples will be used to illustrate the basic concepts discussed in this paper. Since accelerometers are often used in structural system identification and control, a direct transmission term is included in input-output models that are discussed in this paper. Minor adjustments can be easily made when this term is not present which is the case treated in Ref. 1.

2. The Neural Networks

A neural network is simply a set of interconnected individual units called neurons. Depending on the connection between the neurons, there are two basic types of networks known as the multi-layer feedforward networks and the recurrent networks, which will be described in this section.

2.1 The Neuron. As a basic building block for a neural network, an individual neuron has a finite number of scalar inputs and one scalar output. Associated with each input is a scalar weighting value. The input signals are weighted by these values and added at the summation junction. The combined signal is then passed through an activation function producing the output signal. The activation function $\gamma(x)$ can take a variety of forms, the most common one is a sigmoid function denoted by $sigm(x)$,

$$sigm(x) = \frac{1-e^{-x}}{1+e^{-x}} \tag{1}$$

A plot of the sigmoid function is shown in Fig. 1 below. Generally, the activation function can be any non-decreasing differentiable function which has finite limits at both ends as shown in Fig. 1 below.

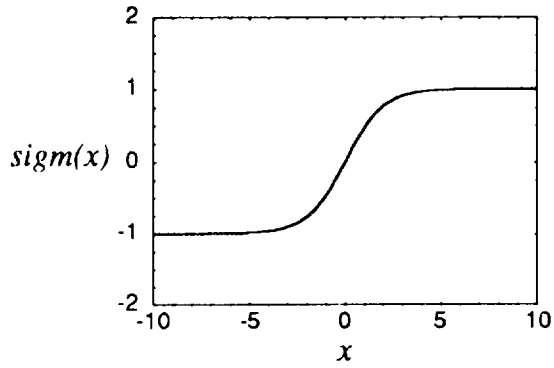

Figure 1: The sigmoid function.

Let r inputs of a neuron be denoted by u_1, u_2, ..., u_r and the output denoted by y. Let the r weights be denoted by w_1, w_2, ..., w_r. The output of the neuron can be expressed mathematically as

$$y = \gamma(x) \ , \quad x = \sum_{i=1}^{r} w_i u_i \tag{2}$$

The neuron is shown schematically in Fig. 2 below with the sigmoid function as the activation function. For simplicity of notations, the network weights for the i-th layer may sometimes be presented collectively as $W_i = \{w_1, w_2, w_3, \ldots\}$.

2.3 Recurrent Network. A feedforward network with time delay feedback elements is called a recurrent network. The delay elements take the outputs of certain neurons in the network, delay them for a certain number of time steps, and feed back as input to the neurons. In other words, in a recurrent network, time delayed outputs of a certain number of neurons are the inputs to other neurons. A special one layer network where the delayed outputs of the neurons are fed back as inputs to themselves is called a Hopfield network (see Fig. 4).

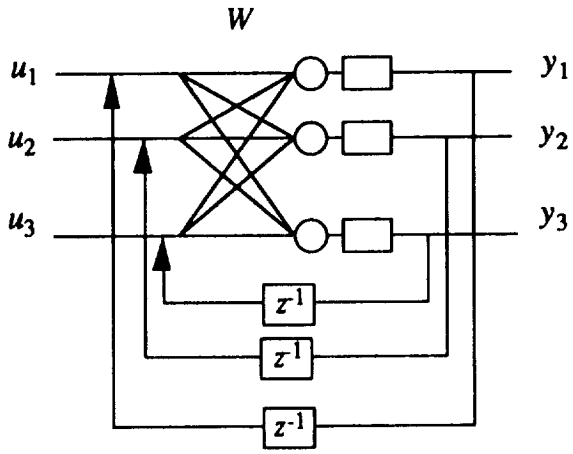

Figure 4: A three-input three-output Hopfield network.

Remark 2.3.1. Unlike a feedforward network, a recurrent network contains self-propagation dynamics. Due to the feedback mechanism, starting with some non-zero initial conditions, the output values of a recurrent network will evolve over time, thus simulating a dynamic system.

Remark 2.3.2. When compared to a recurrent network, a feedforward network is a static in the sense that it simply accepts a set of input values (or input pattern) and produces a set of output values (or output pattern) without any self-propagation mechanism. Thus it may seem that a recurrent network is preferred for modelling dynamic systems. In fact, the two types of networks simply represent two different ways of modelling a dynamic system. In identification, if a set of input-output data is already available, then the weights of a recurrent network can be identified by a feedforward network where the time-delayed outputs are treated as inputs. This seems a bit confusing, but should later be obvious when connections between these types of neural networks to standard ways of representing linear systems are made in the next section.

3. Neural Network Representation of Linear Systems

For each neuron, the only element that is non-linear is the activation function. If the activation is taken to be a linear function, $\gamma(x) = x$, then the network becomes linear. This is true no matter how complicated the neural network is. In this section, attention will be focused on linear networks. The relationship between these networks and conventional ways to represent linear systems will be discussed.

3.1 A Multi-Layer Feedforward Linear Network. Consider the feedforward network shown in Figure 3, with the activation function being a linear function. Furthermore, it is instructive to examine a simple case of a two-layer three-input one-output network shown in Figure 4 below.

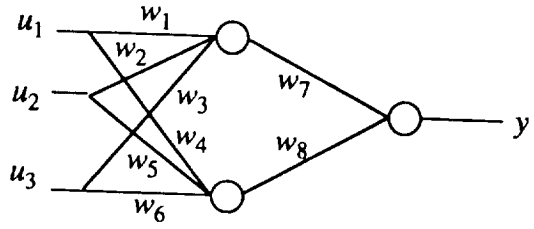

Figure 5: A two-layer three-input one-output feedforward network of linear neurons

The network weights between the individual connections are shown in the figure. Since the neurons are linear, each neuron is represented by a summation junction and the activation being a linear function is omitted. The output of the network in Fig. 5 is simply,

$$\begin{aligned} y &= w_7(w_1u_1 + w_2u_2 + w_3u_3) + w_8(w_4u_1 + w_5u_2 + w_6u_3) \\ &= (w_7w_1 + w_8w_4)u_1 + (w_7w_2 + w_8w_5)u_2 + (w_7w_3 + w_8w_6)u_3 \\ &= \overline{w}_1u_1 + \overline{w}_2u_2 + \overline{w}_3u_3 \end{aligned} \tag{5}$$

which is the output of a single neuron with the following weights:

$$\begin{aligned} \overline{w}_1 &\equiv w_7w_1 + w_8w_4 \\ \overline{w}_2 &\equiv w_7w_2 + w_8w_5 \\ \overline{w}_3 &\equiv w_7w_3 + w_8w_6 \end{aligned} \tag{6}$$

The above example can be immediately generalized to show that a single-output multi-layer neural network with linear neurons is equivalent to a single linear neuron with appropriate weights. The following remarks can be immediately made.

Remark 3.1.1. A multi-layer feedforward network of linear neurons is simply an over-parameterized set of linear equations where the over-parameterization takes the form of the type shown in Eqs. (6). This form is non-linear in the parameters. Thus the problem of determining these parameters from known input-output data is a non-linear parameter estimation problem even if the network is linear.

Remark 3.1.2. Since any single-output feedforward network of linear neurons is equivalent to a particular single linear neuron, there is no benefit in using an over-parameterized multi-layer linear network for linear system identification. In fact, in such an over-parameterized model, the network weights cannot be uniquely determined from input-output data. This is obvious, for example, in the case of the network shown in Fig. 5. The set of three equations in (6) contains eight unknowns. Thus the use of a complicated multi-layer linear network for linear system identification is therefore neither advantageous nor necessary. The same is true for the case of using a non-linear network to identify a linear system.

Remark 3.1.3. For a multi-output system, a multi-layer feedforward network of linear neurons is equivalent to a collection of single neurons arranged in parallel, each of which sharing the same number of inputs. In other words, any multi-output multi-layer feedforward network of linear neurons has an equivalent multi-output single-layer network representation.

3.2. Feedforward Linear Network and the State Space Model. This section describes the relationship between the feedforward linear network and the state space model which is a common form of representing linear systems. The discrete-time state space model of an n-th order, m-input, q-output system is a set of n simultaneous first order difference equations of the form

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \tag{7}$$

where the dimensions of A, B, C, and D are $n \times n$, $n \times m$, $q \times n$, and $q \times m$, respectively.

Solving for the output $y(k)$ in terms of the previous inputs yields

$$y(k) = \sum_{i=0}^{k} h_i u(k-i) \tag{8}$$

where the parameters,

$$h_0 = D \ , \quad h_k = CA^{k-1}B \quad , \quad k = 1,\ 2,\ 3,\ \ldots \tag{9}$$

are the *Markov parameters* of the system described by Eqs. (7), which are also the system pulse response samples. The Markov parameters are expressed in terms of the system discrete state space matrices A, B, C, D. Since the state vector is coordinate-dependent, the state space matrices are not unique for a given system but the Markov parameters are unique. Let the state vector be transformed by a coordinate transformation T, $z(k) = Tx(k)$, then the relationship between $u(k)$ and $y(k)$ via a new state vector $z(k)$ can be described by a new state space representation TAT^{-1}, TB, CT^{-1}, D. The system Markov parameters computed using the new state space matrices are the same as before, i.e.,

$$h_k = CT^{-1}(TAT^{-1})^{k-1}TB = CA^{k-1}B \tag{10}$$

For an asymptotically stable systems, the pulse response can be neglected after a finite number of time steps, say p_s. The input-output description in Eq. (8) can be approximated by a finite number of Markov parameters

$$y(k) \approx h_0 u(k) + h_1 u(k-1) + h_2 u(k-2) \ + \ \cdots \ + h_{p_s} u(k-p_s) \tag{11}$$

where p_s is sufficiently large such that $CA^kB \approx 0$, $k \geq p_s$. Comparing Eq. (11) with the structure of the linear neurons immediately leads to the following remarks.

Remark 3.2.1. The elements of the Markov parameters are simply the weights of a single-layer linear network where inputs to the network include both current and past values of the input signal. Note that the time delayed inputs do not affect the neuron configuration because they are feedforward signals and thus can be treated as separate input channels. This case is shown in Fig. 6.

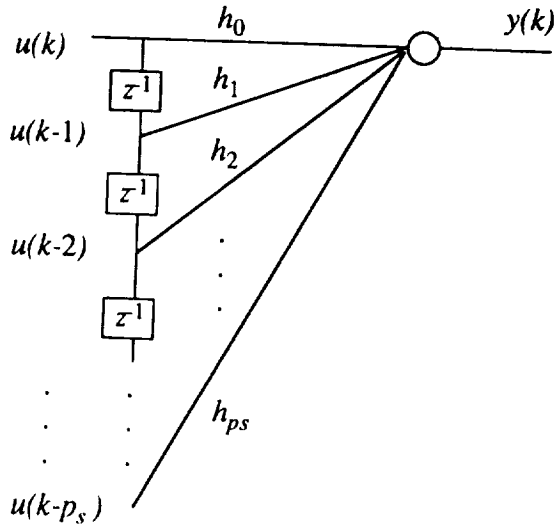

Figure 6: Representation of linear systems by a feedforward network with the system Markov parameters as network weights.

Remark 3.2.2. Since a general multi-layer feedforward network of linear neurons is equivalent to a single-layer network, the relationship between the weights of the multi-layer network which represents a linear system and the Markov parameters of its equivalent state space model is also immediately obvious. For example, if the network shown in Fig. 5 represents some linear system with three non-zero Markov parameters, then

$$\begin{aligned} w_1 &= w_{11}^{(2)} w_{11}^{(1)} + w_{21}^{(2)} w_{12}^{(1)} = h_0 \\ w_2 &= w_{11}^{(2)} w_{21}^{(1)} + w_{21}^{(2)} w_{22}^{(1)} = h_1 \\ w_3 &= w_{11}^{(2)} w_{31}^{(1)} + w_{21}^{(2)} w_{32}^{(1)} = h_2 \end{aligned} \tag{12}$$

provided that $u_1 = u(k)$, $u_2 = u(k-1)$, $u_3 = u(k-2)$.

Remark 3.2.3. In practice, if the system is lightly damped, a large number of system Markov parameters is needed to maintain Eq. (11) a valid approximation. This implies that the equivalent network representing the same system has a large number of input channels containing distant past input values, not a large number of hidden layers. In other words, it is not possible to represent such a system by simply adding extra neurons or extra hidden layers in the feedforward network. The fact that a large number of system Markov parameters is required to represent a lightly damped system of the form in Eq. (11) is a major weakness of the representation. The same can be said for the equivalent neural network representation.

3.2. Recurrent Linear Network and the Observer Model. This section shows the connection between the recurrent network and an observer of the system. Adding and subtracting the term $My(i)$ to the right hand side of the state equation in Eq. (7) yields

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + My(k) - My(k) \\ &= (A + MC)x(k) + (B + MD)u(k) - My(k) \end{aligned} \tag{13}$$

If M is a matrix such that $A + MC$ is deadbeat of order p, i.e.,

$$(A + MC)^k \equiv 0 \ , \quad k \geq p \tag{14}$$

then for $k \geq p$, the output $y(k)$ can be expressed as

$$y(k) = \alpha_1 y(k-1) + \cdots + \alpha_p y(k-p) + \beta_0 u(k) + \beta_1 u(k-1) + \cdots + \beta_p u(k-p) \tag{15}$$

where

$$\begin{aligned} \alpha_k &= -C(A + MC)^{k-1} M \\ \beta_k &= C(A + MC)^{k-1}(B + MD) \ , \quad \beta_0 = h_0 = D \end{aligned} \tag{16}$$

The matrix M in the above development can be interpreted as an observer gain. The system considered in Eqs. (7) has an observer of the form

$$\begin{aligned} \hat{x}(k+1) &= A\hat{x}(k) + Bu(k) - M[y(k) - \hat{y}(k)] \\ \hat{y}(k) &= C\hat{x}(k) + Du(k) \end{aligned} \tag{17}$$

Besides the effect of noises, $\hat{y}(k)$ may differ from $y(k)$ if the actual initial condition $x(0)$ is not known and some different initial condition is assumed for $\hat{x}(0)$. Defining the state estimation error $e(k) = x(k) - \hat{x}(k)$, the equation that governs $e(k)$ is

$$e(k+1) = (A + MC)e(k) \tag{18}$$

For an observable system, the matrix M exists such that the eigenvalues of $A + MC$ may be placed in any desired (symmetric) configuration. If the matrix M is such that $A + MC$ is asymptotically stable, then the estimated state $\hat{x}(k)$ tends to the true state $x(k)$ as k tends to infinity for any initial difference between the assumed observer state and the actual system state. The matrix M can therefore be interpreted as an observer gain. The parameters defined as

$$\begin{aligned} \overline{Y}(k) &= C(A + MC)^{k-1}[B + MD \ , \ -M] \\ &= [\beta_k \ , \ \alpha_k] \end{aligned} \tag{19}$$

are the Markov parameters of an observer system, hence they are referred to as *observer Markov parameters*. Like the system Markov parameters, the observer Markov parameters are also invariant with respect to a coordinate transformation of the state vector. To see this, again let the state vector be transformed by a coordinate transformation T, $z(k) = Tx(k)$, then the observer is described by a new state space representation TAT^{-1}, TB, CT^{-1}, D and a new observer gain TM. The observer Markov parameters computed using these new state space matrices and the new observer gain are the same as before,

$$\bar{Y}(k) = CT^{-1}(TAT^{-1} + TMCT^{-1})^{k-1}[TB + TMD \ , \ -TM]$$
$$= C(A + MC)^{k-1}[B + MD \ , \ -M] \ , \quad k = 1, \ 2, \ 3, \ \ldots \tag{20}$$

Notice that in Eq. (15), the output $y(k)$ is the open-loop response of the system, yet the coefficients α_k, β_k are related to an observer gain. Consider the special case where M is a deadbeat observer gain where all eigenvalues of $A + MC$ are zero, the observer Markov parameters will become identically zero after a finite number of terms. For lightly damped structures, this means that the system can be described by a reduced number of observer Markov parameters $\bar{Y}(k)$ instead of an otherwise large number of the usual system Markov parameters $Y(k)$. For this reason, the observer Markov parameters are important in linear system identification. By examining of the structure of Eq. (15), the following remarks can be made.

Remark 3.3.1. The input-output equation given in Eq. (15) can be represented by a recurrent network with a single layer of linear neurons. The number of neurons is equal to the number of outputs of the system. The inputs to the neurons consists of both the feedforward time-delayed input signals and the feedback time-delayed output signals. Figure 7 shows the configuration of such a network for a single-output system.

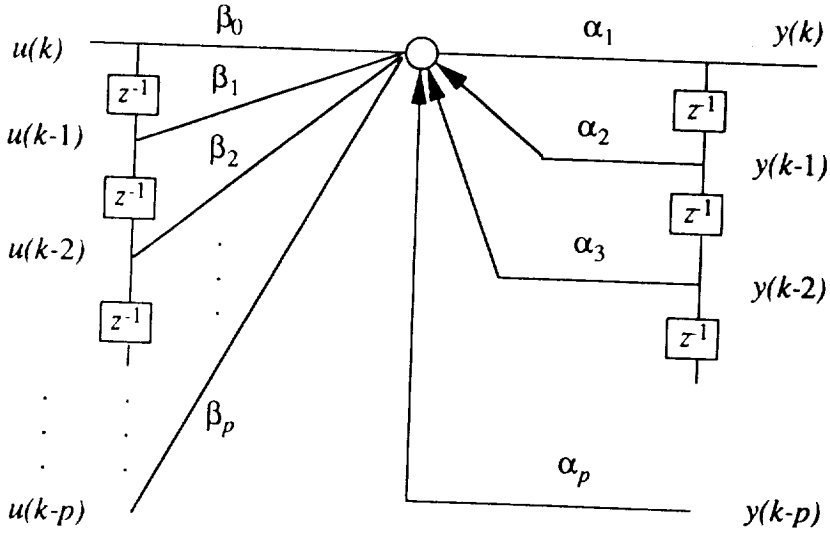

Figure 7: Representation of a single-output system by a recurrent network

Remark 3.3.2. The recurrent network weights are precisely the elements of the observer Markov parameters. The relationship between the weights of a recurrent network and an equivalent feedforward network is the same as that between the observer Markov parameters and the system Markov parameters. It can be shown that the system Markov parameters or the feedforward network weights are related to the recurrent network weights by

$$h_k = \beta_k + \sum_{i=1}^{k} \alpha_i h_{k-i} \tag{21}$$

where $\alpha_k \equiv 0$, $\beta_k \equiv 0$ for $k > p$. To describe a system of order n, the number of observer Markov parameters p must be such that $qp \geq n$ where q is the number of outputs. Furthermore, the maximum order of a system that can be described with p observer Markov parameters is qp.[9] The implication of this result to the network configuration is that a recurrent network generally requires fewer number of parameters (or weights) than that required by an equivalent feedforward network. The two equivalent networks, however, have the same number of neurons. The minimum number of recurrent network weight matrices that can describe the system is $p_{\min}$, which is the smallest value of p such that $qp_{\min} \geq n$.

Remark 3.3.3. As mentioned previously, to represent lightly damped structures, the feedforward representation requires a large number of weights. Furthermore, it is not possible to represent a marginally stable or unstable system by a feedforward network. However, it is possible to represent such a system by a recurrent network. The implication of this fact for the system identification problem will be discussed further in later sections.

4. Identification of Linear Systems using Neural Networks

It has been shown that a general network of linear neurons is equivalent to a single neuron with appropriate weights. The problem of linear system identification using neural network is therefore reduced to finding these network weights from input-output data. The computation may be done off-line or on-line. In off-line computation, the input-output data is already available and a network representing the system is to be determined. On-line computation refers to the case where the network weights are continually updated as data is made available.

4.1. Parallel vs. Series-Parallel Identification Models. In previous consideration, it appears that the recurrent network is more advantageous in representing certain systems than the feedforward network. To identify the recurrent network weights one can simply use the feedforward network configuration with *actual* delayed system outputs appeared as inputs to the feedforward network. Consider two identification models shown in Figs. 8 and 9 below, which are known as parallel and series-parallel identification model. The block denoted by D represents the time delay elements.

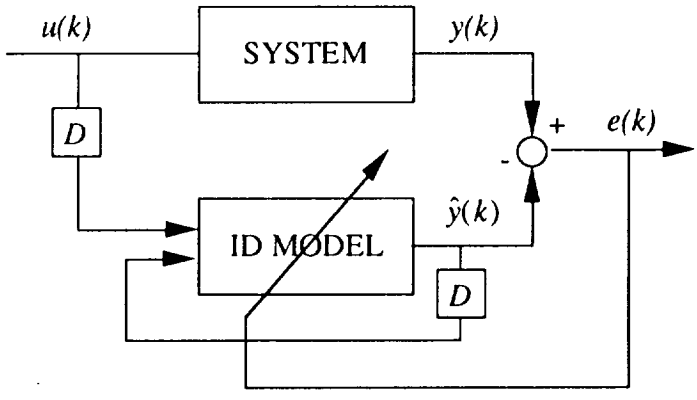

Figure 8: Identification using parallel model.

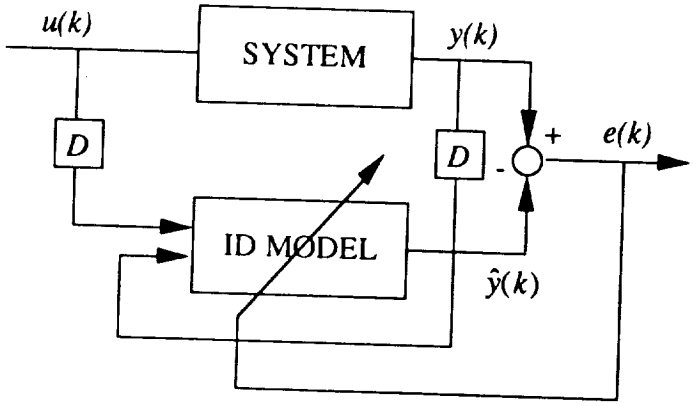

Figure 9: Identification using series-parallel model.

The basic difference between the two schemes is that in the parallel identification model, the estimated output $\hat{y}(k)$ is computed based on the model own previous (estimated) values whereas in the series-parallel model, it is based on actual output values. Mathematically, in the parallel model, the purpose of the identification is to obtain the estimates $\hat{\alpha}_k$, $\hat{\beta}_k$ of the coefficients α_k, β_k that minimize the estimation error, $e(k) = y(k) - \hat{y}(k)$, where the estimated output $\hat{y}(k)$ is computed from

$$\hat{y}(k) = \hat{\alpha}_1\hat{y}(k-1) + \cdots + \hat{\alpha}_p\hat{y}(k-p) + \hat{\beta}_0u(k) + \hat{\beta}_1u(k-1) + \cdots + \hat{\beta}_pu(k-p) \quad (22)$$

In the series-parallel model, however, the estimated output is computed from

$$\hat{y}(k) = \hat{\alpha}_1y(k-1) + \cdots + \hat{\alpha}_py(k-p) + \hat{\beta}_0u(k) + \hat{\beta}_1u(k-1) + \cdots + \hat{\beta}_pu(k-p) \quad (23)$$

The difference between the two above equations is a subtle but important one. As discussed in the previous section, the estimated output of the model in Eq. (22) is the estimated open-loop prediction even though the coefficients of the model are related to an observer. On the other hand, the estimated output of the model in Eq. (23) is that of an observer. To see this, substitute the expression for $\hat{y}(k)$ to the estimated state equation in (17) produces

$$\hat{x}(k+1) = (A + MC)\hat{x}(k) + (B + MD)u(k) - My(k) \quad (24)$$

Since $\hat{y}(k) = C\hat{x}(k) + Du(k)$, one can obtain Eq. (23) assuming zero initial conditions for the observer. Therefore, $\hat{y}(k)$ in Eq. (23) represents the estimated output provided by the observer. The estimation error $\hat{e}(k)$ is the difference between the actual output and the estimated output provided by the observer. On the other hand, if the actual response $y(k)$ is replaced by the estimated value $\hat{y}(k)$ in Eq. (23) then the terms involving the observer gain M cancel each other identically for any arbitrary initial condition $\hat{x}(0)$,

$$\begin{aligned}\hat{x}(k+1) &= (A + MC)\hat{x}(k) + (B + MD)u(k) - M\hat{y}(k) \\ &= A\hat{x}(k) + Bu(k)\end{aligned}$$

Therefore, there is no longer any observer involved in the equation; $\hat{x}(k)$ now plays the role of the state vector $x(k)$ as in Eq. (7) and the estimated output $\hat{y}(k) = C\hat{x}(k) + Du(k)$ is the same as that produced by the open-loop model provided that the initial conditions for $\hat{x}(k)$ and $x(k)$ are identical. The quantity $\hat{y}(k)$ now represents the predicted output provided by the open-loop model alone, which is referred in this paper as open-loop prediction. In this case, the error $\hat{e}(k)$ is the difference between the actual output and the predicted output provided by the identified open-loop model.

Remark 4.1.1. First recall that the model structure in Eq. (15) subsumes an observer. If the parallel identification model is used in conjunction with the model structure of Eq. (15) then the prediction error that drives the parameter estimation scheme is simply the open-loop prediction error not the observer (output) estimation error. Consequently, the observer portion of the model cannot be identified. This fact accounts for the difficulties encountered in parallel model identification, namely, the conditions for which the scheme will converge are presently not known.

Remark 4.1.2. In the series-parallel identification model, since the actual instead of (open-loop) predicted output enters the model, a feedforward network with delayed input and actual output measurements can be used to identify the system. This consideration eliminates the use of a recurrent network which would introduce additional but unnecessary difficulties to the system identification problem, (see Fig. 10). Each output of the system is represented by a single linear neuron. A multiple-output system is represented by a single layer of neurons. The identified network can be used either as a feedforward or a recurrent network. In the former case, the network provides estimation of the response by an observer. In the latter case, it is an open-loop predictor. Again, this depends on whether actual or predicted output is used in computing the response.

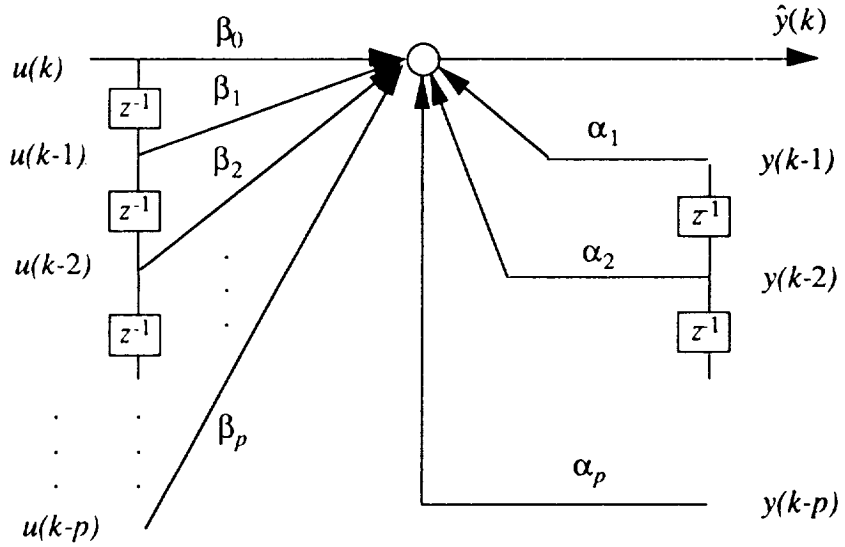

Figure 10: Feedforward representation of the series-parallel identification model by a single neuron for each output.

4.2. Identification of the Network Weights. This section shows how the weights of the network represented by Eq. (15) can be computed using a feedforward model. For linear systems, it is sufficient to use a one layer network having as many neurons as the number of outputs. This is a simple linear parameter estimation problem. The off-line computation is shown first, followed by an equivalent on-line computation.

For simplicity, consider the case where the system starts from zero initial conditions. Equation (15) can be written as

$$y(k) = \sum_{i=1}^{p} [\beta_i \ , \ \alpha_i] \begin{bmatrix} u(k-i) \\ y(k-i) \end{bmatrix} + \beta_0 u(k) \tag{26}$$

where network weight matrices β_i , α_i are defined in Eq. (16). Writing Eq. (26) in matrix form for a set of input-output data $N+1$ samples long yields

$$\mathbf{y} = \mathbf{Y}\mathbf{V} \tag{27}$$

where

$$\mathbf{y} = [y(0) \quad y(1) \quad \cdots \quad y(p) \quad y(p+1) \quad \cdots \quad y(N)] \tag{28}$$

$$\mathbf{Y} = [\beta_0, \ \beta_1, \ \alpha_1, \ \beta_2, \ \alpha_2, \ ..., \ \beta_p, \ \alpha_p] \tag{29}$$

$$\mathbf{V} = \begin{bmatrix} u(0) & u(1) & \cdots & u(p) & u(p+1) & \cdots & u(N) \\ & \begin{bmatrix} u(0) \\ y(0) \end{bmatrix} & \cdots & \begin{bmatrix} u(p-1) \\ y(p-1) \end{bmatrix} & \begin{bmatrix} u(p) \\ y(p) \end{bmatrix} & \cdots & \begin{bmatrix} u(N-1) \\ y(N-1) \end{bmatrix} \\ & & \ddots & \vdots & \vdots & \vdots & \vdots \\ & & & \begin{bmatrix} u(0) \\ y(0) \end{bmatrix} & \begin{bmatrix} u(1) \\ y(1) \end{bmatrix} & \cdots & \begin{bmatrix} u(N-p) \\ y(N-p) \end{bmatrix} \end{bmatrix} \tag{30}$$

The network weight matrices are estimated using the equation

$$\hat{\mathbf{Y}} = \mathbf{y}\mathbf{V}^{+} \tag{31}$$

where $(.)^{+}$ denotes the pseudo-inverse of the quantity in the parentheses. If the initial conditions are not zero then a slightly different equation must be used to solve for the network weights, that is

$$\hat{\mathbf{Y}} = \mathbf{y}_t \mathbf{V}_t^{+} \tag{32}$$

where $\mathbf{y}_t$ and $\mathbf{V}_t$ are obtained by deleting the first p columns in $\mathbf{y}$ and $\mathbf{V}$, respectively.

Remark 4.2.1. The least-squares solution in Eq. (31) or (32) minimizes the error between the actual output and the estimated output computed using the actual input and output data, i.e., the least-squares solution minimizes the residual $\hat{\mathbf{e}} = \mathbf{y} - \hat{\mathbf{y}}$ where $\hat{\mathbf{y}}$ is computed from $\hat{\mathbf{y}} = \hat{\mathbf{Y}}\mathbf{V}$ and $\hat{\mathbf{Y}}$ is given in Eq. (31). If Eq. (32) is used instead, then the least-squares solution minimizes $\hat{\mathbf{e}}_t = \mathbf{y} - \hat{\mathbf{y}}_t$ where $\hat{\mathbf{y}}_t = \hat{\mathbf{Y}}\mathbf{V}_t$. This computation, therefore, corresponds to the series-parallel identification scheme that minimizes the observer estimation error.

Remark 4.2.2. Ideally, the error between the actual output and the predicted output provided by the identified open-loop model is the proper error to be minimized for the identification of the system open-loop model. The above computation minimizes the observer estimation error instead. For a linear system, it turns out that in the absence of

noises the open-loop system can be identified exactly by minimizing the observer estimation error. In the presence of noises, however, minimizing the observer estimation error does not necessarily implies that the open-loop prediction error is minimized. Therefore, it is possible that the observer model fits the data well but the open-loop model does not. Fortunately, if the order of the regression equation is chosen to be sufficiently large then simultaneous observer and system identification will still be achieved in the limit as the data record tends to infinity and the noises are white, Gaussian, and zero-mean (see Ref. 9).

Remark 4.2.3. The least-squares solution in Eq. (31) can be obtained by an on-line parameter estimation scheme. First, write each column in $\mathbf{V}$ as

$$\mathbf{V} = [\Gamma(0),\ \Gamma(1),\ \Gamma(2),\ \ldots\ ] \tag{33}$$

so that at each time step k, Eq. (27) can be written as

$$y(k) = \mathbf{Y}\Gamma(k) \tag{34}$$

The recursive least-squares equation for the network weights is simply,

$$\hat{\mathbf{Y}}(k) = \hat{\mathbf{Y}}(k-1) + \left[y(k) - \hat{\mathbf{Y}}(k-1)\Gamma(k)\right]\left\{\frac{\Gamma(k)^T R(k-1)}{1+\Gamma(k)^T R(k-1)\Gamma(k)}\right\} \tag{35}$$

$$R(k) = R(k-1) - \frac{R(k-1)\Gamma(k)\Gamma(k)^T R(k-1)}{1+\Gamma(k)^T R(k-1)\Gamma(k)} \tag{36}$$

where $\hat{\mathbf{Y}}(k) = \left[\hat{\beta}_0(k),\ \hat{\beta}_1(k),\ \hat{\alpha}_1(k),\ \hat{\beta}_2(k),\ \hat{\alpha}_2(k),\ \ldots.\ \hat{\beta}_p(k),\ \hat{\alpha}_p(k)\right]$, $\hat{\mathbf{Y}}(0)$ is an arbitrary initial guess, and $R(0)$ is any symmetric positive definite matrix. The recursive equations for (32) are analogous.

Remark 4.2.4. In a multi-layer neural network, the back propagation algorithm is typically used to update the network weights recursively. This is a gradient-based parameter update algorithm. When expressed in the block diagram form for hardware implementation, the algorithm resembles the forward network except that the signal travels in the opposite direction, leading to the name back propagation. In the present case, because the network is simply one layer of linear neurons, it is not efficient to use the back propagation algorithm to compute the network weights. For on-line implementation, the least squares algorithm given in Eqs. (35-36) or its variants for fast computation are preferred.

5. Predictor Models For Linear Dynamic Systems

In theory, the open-loop model can be used to predict the system response based on current and past input values. However, this is not desirable in practice because of the requirement that both the open-loop model and the initial conditions be known exactly. Such a prediction is also sensitive to noises. On the other hand, an observer which is typically used to estimate the system state based on actual input-output data can also be used to provide an estimate of the system output. This section first discusses the use of an observer as an one-step ahead predictor. This interpretation is important because of its connection to the control problem which will be discussed in later sections. Extensions to the identification and use of multiple-step ahead predictors will then be made. For linear systems, these predictors are simply special single-layer networks of linear neurons.

5.1. One-Step Ahead Predictor. To express explicitly the observer as an one-step ahead predictor, one simply writes the observer equations as

$$\begin{aligned}\hat{x}(k+1) &= (A+MC)\hat{x}(k)+(B+MD)u(k)-My(k)\\ \hat{y}(k+1) &= C\hat{x}(k+1)+Du(k+1)\end{aligned} \tag{37}$$

As a predictor, the interested quantity is $\hat{y}(k+1)$. One can therefore bypass the state equation by writing

$$\hat{y}(k+1) = \alpha_1 y(k)+ \cdots +\alpha_p y(k-p+1)+\beta_0 u(k+1)+\beta_1 u(k)+ \cdots +\beta_p u(k-p+1) \tag{38}$$

The following remarks can be made regarding the forms of Eq. (37) and Eq. (38).

Remark 5.1.1. In theory, if the state space model (A, B, C, D) is known exactly then one can design an observer gain M such that $A+MC$ is asymptotically stable. To use Eq. (38) as an output predictor, one need to include a sufficient number of terms such that $(A+MC)^i$ is negligible for $i \geq p$. The state space representation in Eq. (37) is a better choice since it involves no such approximation. The above comment no longer holds true if M is such that $A+MC$ is deadbeat, i.e., $(A+MC)^i \equiv 0$, $i \geq p$ since the approximation becomes exact in this case.

Remark 5.1.2. In practice, the system model cannot be known exactly. To identify the system from input-output data using the series-parallel structure, one in fact computes directly the coefficients in Eq. (38) rather than the state space matrices. To obtain a minimal order state space representation from these coefficients, realization is required. As an output predictor, therefore, Eq. (38) should be used directly because conversion to a state space representation is not necessary for this purpose.

Remark 5.1.3. Equation (38) clearly indicates that the one-step ahead predictor takes the form of a single layer network of linear neurons with actual input and output signals entering the network, and the output of the network represents the one-step ahead prediction. Schematically, this is the same as shown in Fig. 10.

5.2. A Two-Step Ahead Predictor. This section derives the equations for a two-step ahead predictor for linear systems, and shows that it also has a linear neural network form. First, from Eq. (7), one can write

$$\begin{aligned}x(k+2) &= Ax(k+1)+Bu(k+1)\\ &= A^2x(k)+ABu(k)+Bu(k+1)\\ y(k+2) &= Cx(k+2)+Du(k+2)\end{aligned} \tag{39}$$

Adding and subtracting the term $Gy(i)$ to the right hand side of the state equation yields

$$\begin{aligned}x(k+2) &= A^2x(k)+ABu(k)+Bu(k+1)+Gy(k)-Gy(k)\\ &= (A^2+GC)x(k)+\begin{bmatrix}AB+GD, & B\end{bmatrix}\begin{bmatrix}u(k)\\ u(k+1)\end{bmatrix}-Gy(k)\end{aligned} \tag{40}$$

If G is a matrix such that A^2+GC is deadbeat of order p, i.e.,

$$(A^2 + GC)^k \equiv 0 \ , \quad k \geq p \tag{41}$$

then the relationship between the input and output of the system can be described as a linear combination of input-output data of the form

$$y(k+2) = g(y(k),\ y(k-2),\ y(k-4),\ ...,\ u(k+2),\ u(k+1),\ u(k),\ ...) \tag{42}$$

for $k \geq 2p-2$ so that at sufficiently large time steps, terms involving the states $x(0)$ and $x(1)$ vanish, i.e., $C(A^2+GC)^i x(0) = 0$, $C(A^2+GC)^i x(1) = 0$, for $i \geq p$ due to the imposed deadbeat condition for $A^2 + GC$. Furthermore, there is only a finite number of coefficients that make up the linear combination in $g(.)$, which are the *predictor Markov parameters* of the form

$$\overline{G}(k) = C(A^2+GC)^{k-1}\left[AB+GD,\ B\ ,\ -G\right]\ ,\ k = 1,\ 2,\ ...,\ p \tag{43}$$

Existence of the matrix G such that $(A^2 + GC)^k \equiv 0$, $k \geq p$ is assured if the pair $(A^2,\ C)$ is observable.

Remark 5.2.1. The above derivation justifies the form of a two-step ahead predictor. In fact, one can identify the coefficients of this predictor from input-output data by minimizing the two-step prediction error. The procedure is similar to that discussed in Section 4.

Remark 5.2.2. To obtain the two-step ahead prediction one can also propagate the observer, which is a one-step ahead predictor, in two successive time steps by treating the estimated output from the first time step as the actual output for the second time step However, such a procedure would amount to performing open-loop prediction in the second time step and is therefore sensitive to noises. On the other hand, if one uses the predictor form with the coefficients directly identified from input-output data then only actual data enter the computation and thus minimizes the errors due to noises.

Remark 5.2.3. Again, the predictor form can be represented by a single layer of linear neurons, and the weights of this network are simply the elements of the predictor Markov parameters shown in Eq. (43). Results presented in this section can be easily generalized to a general multi-step predictor. The relationship between such predictors and the deadbeat control problem will be discussed in a later reference.

6. Control of Linear Systems using Neural Networks

As formulated in previous sections, linear systems can be represented by a single-layer network of linear neurons. The weights of this network can be identified from input-output data. Once identified, the network can be used as a one-step ahead predictor. This section discusses the use of such network directly for control application without requiring the state space model to be extracted from these weights.

6.1. A One-Step Ahead Controller. First, consider the case where the linear system can be expressed in the form,

$$y(k+1) = \alpha_1 y(k) + \ \cdots\ + \alpha_p y(k-p+1) + \beta_0 u(k+1) + \beta_1 u(k) + \ \cdots\ + \beta_p u(k-p+1) \tag{44}$$

where the coefficients are assumed to be known. Let the desired response be denoted by $r(k)$. To obtain a controller directly from the above equation, one simply replaces $y(k+1)$

by its desired value $r(k+1)$ and then solve for the control input $u(k+1)$ to obtain the control law,

$$u(k+1)=\beta_0^{-1}\left[r(k+1)-\alpha_1 y(k)-\cdots-\alpha_p y(k-p+1)-\beta_1 u(k)-\cdots-\beta_p u(k-p+1)\right] \quad (45)$$

If the coefficients in Eq. (44) are not known exactly then Eq. (46) represents an one-step ahead estimate of what the system will produce based on current and past input-output data,

$$\hat{y}(k+1)=\hat{\alpha}_1 y(k)+\cdots+\hat{\alpha}_p y(k-p+1)+\hat{\beta}_0 u(k+1)+\hat{\beta}_1 u(k)+\cdots+\hat{\beta}_p u(k-p+1) \quad (46)$$

The control law then is simply

$$u(k+1)=\hat{\beta}_0^{-1}\left[r(k+1)-\hat{\alpha}_1 y(k)-\cdots-\hat{\alpha}_p y(k-p+1)-\hat{\beta}_1 u(k)-\cdots-\hat{\beta}_p u(k-p+1)\right] \quad (47)$$

The behavior of the closed-loop system using the above control law can be examined as follows. Let $e(k+1)$ denote the error between the actual response and the predicted response,

$$\hat{y}(k+1)=y(k+1)-e(k+1) \quad (48)$$

Substituting Eq. (47) and Eq. (48) into Eq. (46) produces

$$\begin{aligned} y(k+1)-e(k+1) &= \hat{\alpha}_1 y(k)+\cdots+\hat{\alpha}_p y(k-p+1)+\hat{\beta}_1 u(k)+\cdots+\hat{\beta}_p u(k-p+1) \\ &\quad +\hat{\beta}_0\left\{\hat{\beta}_0^{-1}\left[r(k+1)-\hat{\alpha}_1 y(k)-\cdots-\hat{\alpha}_p y(k-p+1)\right.\right. \\ &\quad \left.\left.-\hat{\beta}_1 u(k)-\cdots-\hat{\beta}_p u(k-p+1)\right]\right\} \\ &= r(k+1) \end{aligned} \quad (49)$$

or

$$y(k+1)=r(k+1)+e(k+1) \quad (50)$$

Define the tracking error to be the difference between the actual response and the desired response, $\varepsilon(k)=y(k)-r(k)$, Eq. (50) reveals that

$$\varepsilon(k+1)=e(k+1) \quad (51)$$

Therefore, if the predictor is such that its prediction error vanishes in the limit, then the tracking error also vanishes in the limit, i.e.,

$$\lim_{k\to\infty} e(k)=0 \quad \Rightarrow \quad \lim_{k\to\infty} \varepsilon(k)=0 \quad (52)$$

Remark 6.1.1. The one-step ahead control law has the property that the tracking error is the same as the prediction error. The above analysis shows that accuracy of the predictor model governs the accuracy of the tracking response. As long as the predictor can perform a reasonably good one-step ahead prediction of the system response then the control input can be computed to make the system track a desired trajectory. In the ideal case where the system is linear and the data is noise-free, the prediction error and the

tracking error will be zero identically. Non-zero prediction and tracking error can only come about during adaptation or when noises are present. This is different from the non-linear case where both the estimation and tracking error are non-zero even when there are no noises in the system. An important restriction of the one-step ahead controller is that the open-loop system is required to be stably invertible, (i.e., there are no unstable zeros in the linear case). If this condition is not met, it is possible to have the controller producing unbounded input while maintaining zero tracking error.

Remark 6.1.2. To obtain the result in Eq. (51), the controller coefficients must be the same as those of the predictor model. In the event the coefficients of the predictor model are updated at each time step, then the controller coefficients must also match those of the predictor model. Mathematically, if at time step k, the predictor takes the form

$$\begin{aligned}\hat{y}(k+1) = \hat{\alpha}_1(k)y(k) + \cdots + \hat{\alpha}_p(k)y(k-p+1) \\ + \hat{\beta}_0(k)u(k+1) + \hat{\beta}_1(k)u(k) + \cdots + \hat{\beta}_p(k)u(k-p+1)\end{aligned} \tag{53}$$

then the control law will be taken to be

$$\begin{aligned}u(k+1) = \hat{\beta}_0(k)^{-1}\big[r(k+1) - \hat{\alpha}_1(k)y(k) - \cdots - \hat{\alpha}_p(k)y(k-p+1) \\ - \hat{\beta}_1(k)u(k) - \cdots - \hat{\beta}_p(k)u(k-p+1)\big]\end{aligned} \tag{54}$$

Remark 6.1.3. The above controller can be implemented in neural network form. Such a controller simply copies the weights of the feedforward predictor network to generate the control input. This is shown schematically in Fig. 11 below.

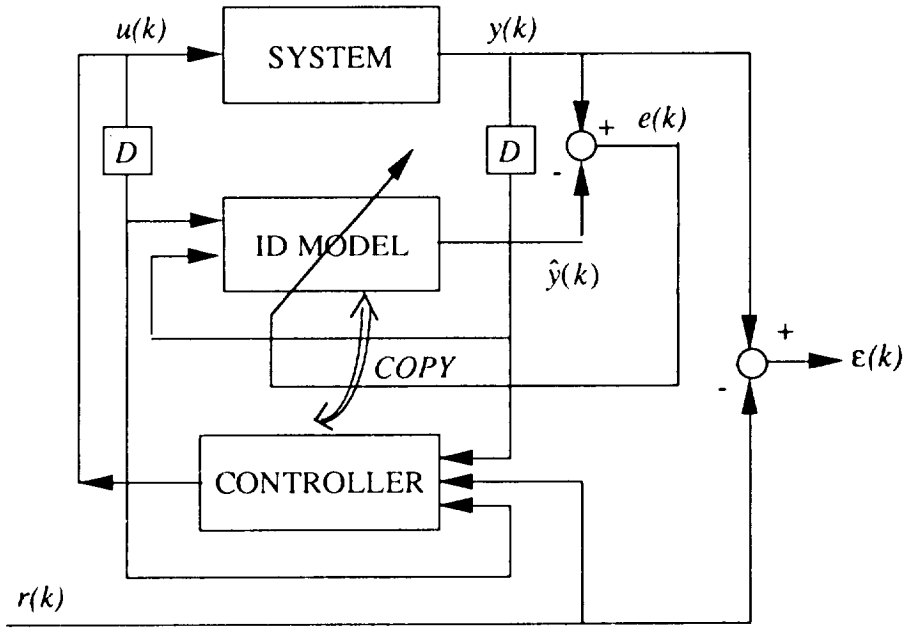

Figure 11: Adaptive one-step ahead controller.

Remark 6.1.4. Since the controller attempts to make the system track the desired trajectory in one step, excessive control efforts are usually required. This makes the approach unattractive in practice. To alleviate this problem, the weighted one-step ahead

controller is used, such that at each time step the control input minimizes the following quadratic cost function

$$J(k+1)=\frac{1}{2}\varepsilon(k+1)^T Q\varepsilon(k+1)+\frac{1}{2}u(k+1)^T Su(k+1) \tag{55}$$

where the tracking error $\varepsilon(k+1)=y(k+1)-r(k+1)$. The weighting matrices Q and S are required to be symmetric and positive definite. Substituting the expression for $\varepsilon(k+1)$ and $y(k+1)$ into the cost function and then performing the minimization produces

$$\begin{aligned}\frac{\partial J(k+1)}{\partial u(k+1)}&=\beta_0^T Q\alpha_1 y(k)+\beta_0^T Q\alpha_2 y(k-1)+\cdots+\beta_0^T Q\alpha_p y(k-p+1)+\beta_0^T Q\beta_0 u(k+1)\\&\quad+\beta_0^T Q\beta_1 u(k)+\cdots+\beta_0^T Q\beta_p u(k-p+1)-\beta_0^T Qr(k+1)+Su(k+1)\end{aligned} \tag{56}$$

Setting the result to zero and solving for the control input yields

$$\begin{aligned}u(k+1)=(\beta_0^T Q\beta_0+S)^{-1}\beta_0^T Q\big[&r(k+1)-\alpha_1 y(k)-\cdots-\alpha_p y(k-p+1)\\&-\beta_1 u(k)-\cdots-\beta_p u(k-p+1)\big]\end{aligned} \tag{57}$$

The above is known as a weighted one-step ahead controller in adaptive control literature.[6]

6.2. Model Reference Controller. A different way to avoid the requirement that the system track a desired trajectory in one step is to use a control scheme known as model reference control. Let the control law in Eq. (47) be modified as

$$\begin{aligned}u(k+1)=\hat{\beta}_0^{-1}\big[&r(k+1)-\hat{\alpha}_1 y(k)-\cdots-\hat{\alpha}_p y(k-p+1)-\hat{\beta}_1 u(k)-\cdots-\hat{\beta}_p u(k-p+1)\\&-\hat{\gamma}_1 y(k)-\cdots-\hat{\gamma}_p y(k-p+1)\big]\end{aligned} \tag{58}$$

Substituting Eq. (48) and Eq. (58) into Eq. (46) yields

$$\begin{aligned}y(k+1)-e(k+1)&=\hat{\alpha}_1 y(k)+\cdots+\hat{\alpha}_p y(k-p+1)+\hat{\beta}_1 u(k)+\cdots+\hat{\beta}_p u(k-p+1)\\&\quad+\hat{\beta}_0\Big\{\hat{\beta}_0^{-1}\big[-\hat{\alpha}_1 y(k)-\cdots-\hat{\alpha}_p y(k-p+1)-\hat{\beta}_1 u(k)-\cdots-\hat{\beta}_p u(k-p+1)\\&\quad-\gamma_1 y(k)-\cdots-\gamma_p y(k-p+1)+r(k+1)\big]\Big\}\\&=-\gamma_1 y(k)-\cdots-\gamma_p y(k-p+1)+r(k+1)\end{aligned} \tag{59}$$

which can be expressed as

$$y(k+1)+\gamma_1 y(k)+\cdots+\gamma_p y(k-p+1)=r(k+1)+e(k+1) \tag{60}$$

The system response $y(k)$ now no longer follows the reference input $r(k)$ directly as in the case in Eq. (49). Its behavior can be conveniently interpreted in terms of a reference model. Define $y_m(k)$ as the response of a reference model when driven by the reference input $r(k)$,

$$y_m(k+1)+\gamma_1 y_m(k)+\cdots+\gamma_p y_m(k-p+1)=r(k+1) \tag{61}$$

and the tracking error $\varepsilon_m(k)$ as the difference between the system response $y(k)$ and the reference model response $y_m(k)$,

$$\varepsilon_m(k)=y(k)-y_m(k) \tag{62}$$

The equation that governs the behavior of this tracking error is obtained by subtracting Eq. (61) from Eq. (60),

$$\varepsilon_m(k+1)+\gamma_1\varepsilon_m(k)+\cdots+\gamma_p\varepsilon_m(k-p+1)=e(k+1) \tag{63}$$

Therefore, convergence of the prediction error to zero implies convergence of the tracking error to zero provided that the characteristic equation governing the homogeneous part of the difference equation is asymptotically stable,

$$\lambda^p+\gamma_1\lambda^{p-1}+\cdots+\gamma_p=0 \tag{64}$$

This requirement is easily satisfied since the coefficients γ_1, γ_2, ..., γ_p are the design variables to be selected a priori.

Remark 6.2.1. The difference between this case and the previous case is that the desired trajectory is not specified by the reference input $r(k)$, but rather by the response of the reference model. Since the reference model is known, the reference input $r(k)$ that is needed to make the reference model produces the desired response can be easily computed. The introduction of the reference model is to slow down the convergence of the tracking error so that excessive correction during the adaptation process does not occur.

Remark 6.2.2. The model reference control scheme can also be implemented in neural network form. At any time step, the controller network copies the coefficients of the predictor network and uses them in the generation of the control input. The configuration for this control scheme is shown in Fig. 12.

Remark 6.2.3. Equation (63) shows that the prediction error acts as a driving term for the difference equation that governs the behavior of the tracking error. If the reference model coefficients are designed such that the homogeneous solution is asymptotically stable then the steady state tracking error is simply the particular solution of the difference equation. One thus has the ability to affect the steady state tracking error through the reference model coefficients. However, this freedom is constrained by the residual dynamics of the prediction error that the steady state tracking error may be amplified or reduced. Generally speaking, the natural frequencies of the reference model should be placed away from those dominating the residual dynamics.

Remark 6.2.4. If the coefficients of the predictor model are updated at each time step, then the controller coefficients must match those of the predictor model at each time step. The resulting integration between parameter estimation and control computation is known as model reference adaptive control. The adaptive scheme is summarized in the following equations where the ordinary least-squares algorithm is used to perform the parameter estimation step. Again, let $\hat{\mathbf{Y}}(k)$ denote the estimated coefficients of the predictor model at time step k,

$$\hat{\mathbf{Y}}(k)=\left[\hat{\beta}_0(k),\ \hat{\beta}_1(k),\ \hat{\alpha}_1(k),\ \hat{\beta}_2(k),\ \hat{\alpha}_2(k),\ \ldots,\ \hat{\beta}_p(k),\ \hat{\alpha}_p(k)\right] \tag{65}$$

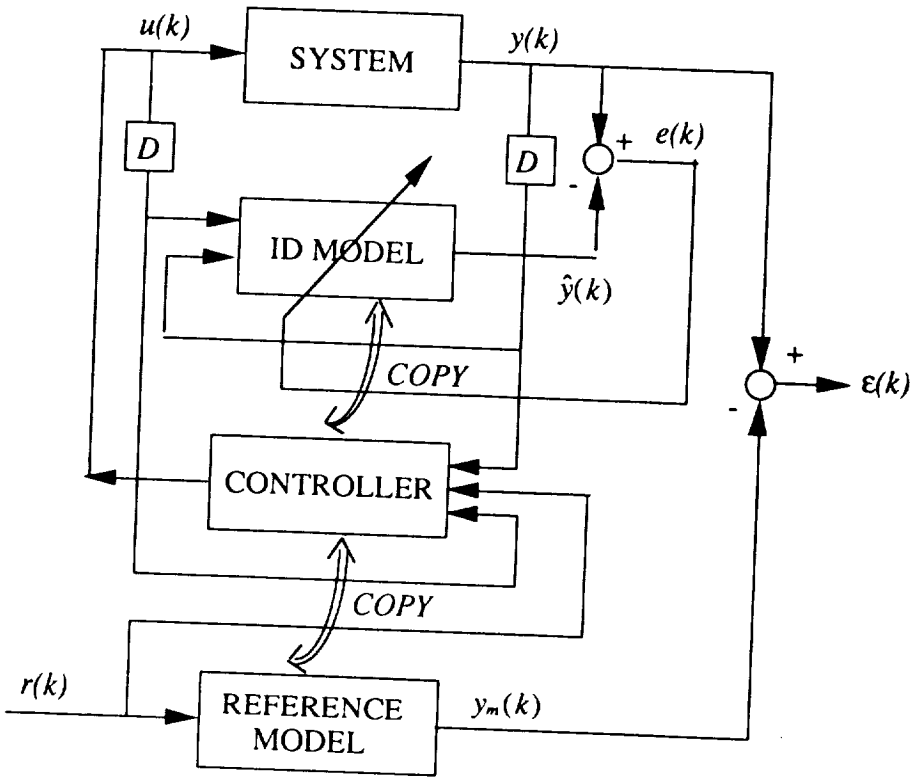

Figure 12: Model reference adaptive control.

starting with $\hat{\mathbf{Y}}(0)$ as an arbitrary initial guess. The control input is computed from

$$u(k+1) = \hat{\beta}_0(k)^{-1}\Big[-\hat{\alpha}_1(k)y(k) - \cdots - \hat{\alpha}_n(k)u(k-n+1) - \cdots - \hat{\beta}_p(k)u(k-p+1) - \gamma_1 y(k) - \cdots - \gamma_p y(k-p+1) + r(k+1)\Big] \tag{66}$$

where the reference model coefficients $\gamma_1, \gamma_2, \ldots, \gamma_p$ are time-invariant and chosen a priori. The above control input is applied to the system producing response $y(k+1)$. The predictor coefficients are then updated according to the rule

$$\hat{\mathbf{Y}}(k+1) = \hat{\mathbf{Y}}(k) + \Big[y(k+1) - \hat{\mathbf{Y}}(k)\Gamma(k+1)\Big]\left\{\frac{\Gamma(k+1)^T R(k)}{1+\Gamma(k+1)^T R(k)\Gamma(k+1)}\right\} \tag{67}$$

$$R(k+1) = R(k) - \frac{R(k)\Gamma(k+1)\Gamma(k+1)^T R(k)}{1+\Gamma(k+1)^T R(k)\Gamma(k+1)} \tag{68}$$

staring with $R(0)$ as any symmetric positive definite matrix. The newly estimated parameters are then used to compute the control input for the next time step $u(k+2)$.

Remark 6.2.5. The control schemes discussed in this section deals with a one-step ahead predictor model of the form shown in Eq. (38). The previous section shows that a two-step ahead predictor or a multi-step ahead predictor has the same linear form. Therefore, the results presented in this section can be easily

For example, the two-step ahead controller will compute the control $u(k+2)$ requiring the measurements upto $y(k)$ only.

7. Modelling and Control of Non-Linear Systems

Up to this point, the discussion has been restricted to linear systems. It has been shown that for linear systems, it is not necessary to have a complicated neural network for identification and control, but rather a single layer of linear neurons is adequate. This section extends the results to non-linear systems. The significance of linear predictor models for non-linear systems will be discussed. This has an important implication on the extent to which linear techniques can be used to handle non-linear systems. The modelling and control of non-linear systems using non-linear neural networks will then be examined.

7.1. Linear Predictor Models for Non-Linear Systems. The predictor model derived in Section 5 is based on the open-loop state space model which is a linear representation. For linear systems, the identified coefficients of an one-step ahead predictor take a particular form, namely, the Markov parameters of an observer model consisting of the open-loop state space model and an observer gain. Recall that a predictor uses actual input and output data to compute the predicted response at each time step. To qualify as a valid open-loop model as well, the predictor must also accurately predicts the system response in an open-loop test using input data alone. As mentioned previously, for linear systems, the predictor is also valid as an open-loop model because it can also produce correct open-loop prediction. For a non-linear system, this is no longer the case. However, when the predicted response is modeled as a linear combination of past input and output data, it turns out that surprisingly good prediction can still be obtained even for non-linear systems. Such predictor models do not qualify as open-loop models of the actual system because they do not predict correct open-loop response using input data alone. This point will be further illustrated by a numerical example in a later section.

7.2. Control of Non-Linear Systems Using Predictor Models. In this section, we show that the model reference controller considered in Section 6.2, or its special version, the one-step ahead controller in Section 6.1, can be used to control a class of non-linear systems which can be represented by linear predictors of the form considered in Eq. (38). Suppose that the non-linear system can be represented by an non-linear auto-regressive model of the form

$$y(k+1) = f\big(y(k),\ y(k-1),\ \ldots,\ u(k+1),\ u(k),\ u(k-1),\ \ldots\big) \tag{69}$$

where $f(.)$ is some non-linear function of past input and output data. First, note that for the response of the system to follow that of a reference model,

$$y_m(k+1) + \gamma_1 y_m(k) + \cdots + \gamma_p y_m(k-p+1) = r(k+1) \tag{70}$$

we require that the response of the controlled system be described by

$$y(k+1) + \gamma_1 y(k) + \cdots + \gamma_p y(k-p+1) = r(k+1) \tag{71}$$

so that the tracking error, $\varepsilon_m(k) = y(k) - y_m(k)$, will be governed by

$$\varepsilon_m(k+1) + \gamma_1 \varepsilon_m(k) + \cdots + \gamma_p \varepsilon_m(k-p+1) = 0 \tag{72}$$

Therefore, at time step $k+1$, one wishes to determine the control input $u(k+1)$ such that Eq. (69) is satisfied. Since the relationship between $y(k+1)$ and $u(k+1)$ is non-linear and is not known, one cannot solve for $u(k+1)$ directly. However, if the non-linear system is such that there exists a predictor of the form given in Eq. (44) such that $\hat{y}(k+1) \approx y(k+1)$ then one satisfies the following equation,

$$\hat{y}(k+1) + \gamma_1 y(k) + \cdots + \gamma_p y(k-p+1) = r(k+1)$$

instead of Eq. (71). The control law is then determined by substituting Eq. (46) in Eq. (73) and then solve for $u(k+1)$, producing exactly the same control law as given in Eq. (66). The fact that the control law satisfies Eq. (73) instead of Eq. (71) will make the tracking error equation governed by Eq. (63) instead of Eq. (72), where $e(k+1)$ denotes the prediction error defined in Eq. (48).

7.3. Identification and Control of Non-Linear Systems using Non-Linear Neural Networks. The identification and control scheme discussed in this paper can be extended to include non-linear neurons. The basic assumption is that the system response is a non-linear function of previous input and output data which can be represented by a multi-layer feedforward network having a sufficient number of non-linear neurons. Let the non-linear function be denoted by $f(.)$ and its neural network representation by $N(.)$,

$$\begin{aligned} y(k+1) &= f(y(k),\ y(k-1),\ \ldots,\ u(k+1),\ u(k),\ u(k-1),\ \ldots) \\ &= N(y(k),\ y(k-1),\ \ldots,\ u(k+1),\ u(k),\ u(k-1),\ \ldots) \end{aligned} \tag{74}$$

When a non-linear network of sufficiently large number of hidden layers is used, then it may also qualify as an open-loop model of the non-linear system besides its being an one-step head predictor. This is the fundamental difference between identification using a linear network versus a non-linear network. Generally speaking, the theoretical advantage of using a non-linear network for non-linear system identification is off-set by the difficulties in finding such a network in practice. Neither the number of hidden layers nor the number of neurons in each layer are known a priori. For a chosen network configuration, the back propagation algorithm is often used to determine the network weights. Typically, the convergence rate is slow and a large amount of data is needed. The back propagation algorithm is well-known and discussed extensively in the literature.

In the model reference control problem, the theoretical advantage of a non-linear network is somewhat diminished because the open-loop model need not be found for purpose of tracking control. The model reference control scheme can accommodate a non-linear network rather easily. Assume that the network representing the non-linear system can be expressed in the form,

$$\begin{aligned} y(k+1) &= N(y(k),\ y(k-1),\ \ldots,\ u(k+1),\ u(k),\ u(k-1),\ \ldots) \\ &= N_1(y(k),\ y(k-1),\ \ldots,\ u(k),\ u(k-1),\ \ldots) + \beta_0(k)u(k+1) + e_1(k+1) \end{aligned} \tag{75}$$

where $e_1(k+1)$ denotes the fitting error introduced with the separation of the $u(k+1)$ term from $N(.)$. The control input is computed from

$$u(k+1) = \beta_0(k)^{-1}\left[r(k+1) - \gamma_1 y(k) - \cdots - \gamma_p y(k-p-1) - N_1(.)\right] \tag{76}$$

where γ_1, γ_2,..., γ_p are the coefficients of the reference model representing the desired response. The control input when applied to the system yields the closed-loop response,

$$\begin{aligned} y(k+1) &= N_1(.) + \beta_0(k)\left\{\beta_0(k)^{-1}\left[r(k+1) - \gamma_1 y(k) - \cdots - \gamma_p y(k-p-1) - N_1(.)\right]\right\} + e_1(k+1) \\ &= N_1(.) + r(k+1) - \gamma_1 y(k) - \cdots - \gamma_p y(k-p-1) - N_1(.) + e_1(k+1) \\ &= r(k+1) - \gamma_1 y(k) - \cdots - \gamma_p y(k-p-1) + e_1(k+1) \end{aligned} \tag{77}$$

The tracking error, $\varepsilon_m(k) = y(k) - y_m(k)$, where $y_m(k)$ is the response of the reference model is governed by the difference equation

$$\varepsilon_m(k+1) + \gamma_1 \varepsilon_m(k) + \cdots + \gamma_p \varepsilon_m(k-p+1) = e_1(k+1) \tag{78}$$

In practice, one identifies an approximation of $N_1(.)$ denoted by $\hat{N}_1(.)$. The control law is then based on $\hat{N}_1(.)$,

$$u(k+1) = \beta_0(k)^{-1}\left[y(k+1) - \gamma_1 y(k) - \cdots - \gamma_p y(k-p-1) - \hat{N}_1(.)\right] \tag{79}$$

The closed-loop system becomes

$$y(k+1) = N_1(.) + r(k+1) - \gamma_1 y(k) - \cdots - \gamma_p y(k-p-1) - \hat{N}_1(.) + e_1(k+1) \tag{80}$$

Let $e_2(k+1)$ denote the approximation error, $e_2(k+1) = N_1(.) - \hat{N}_1(.)$. The tracking error is now governed by

$$\varepsilon_m(k+1) + \gamma_1 \varepsilon_m(k) + \cdots + \gamma_p \varepsilon_m(k-p+1) = e_1(k+1) + e_2(k+1) \tag{81}$$

A schematic diagram of the control scheme is the same as shown in Fig. 12, except that the block representing the identification model is now a non-linear neural network.

8. Numerical Examples

In this section, several examples will be presented to illustrate various concepts discussed in this paper. The case of a linear system is considered first, followed by a non-linear system. Both identification and control aspects of each case will be shown.

8.1. Network Representation of a Linear System. Consider a linear single-input single-output system with three vibration modes at 0.40Hz, 1.37Hz, and 2.21Hz, each with a damping factor of 0.5%. The state space matrices shown represent a discrete model at a sampling rate of 10 Hz.

$$A = \begin{bmatrix} 0.6585 & 0.2100 & 0.0144 & 0.4391 & 0.0382 & 0.0017 \\ 0.2100 & 0.4701 & 0.3149 & 0.0382 & 0.4035 & 0.0573 \\ 0.0144 & 0.3149 & 0.6705 & 0.0017 & 0.0573 & 0.4405 \\ -1.2408 & 0.6925 & 0.1093 & 0.6516 & 0.2119 & 0.0155 \\ 0.6925 & -1.7694 & 1.0387 & 0.2119 & 0.4630 & 0.3179 \\ 0.1093 & 1.0387 & -1.1498 & 0.0155 & 0.3179 & 0.6646 \end{bmatrix}, \quad B = \begin{bmatrix} 0.0050 \\ 0.1124 \\ 0.0075 \\ 0.0382 \\ 0.4035 \\ 0.0573 \end{bmatrix} \tag{82}$$

$$C = [1.0 \quad -0.5 \quad 0.0 \quad 1.0 \quad 0.5 \quad 0.0], \quad D = 1.5$$

The system is excited by random input shown in Fig.13 producing the response shown in Fig.14.

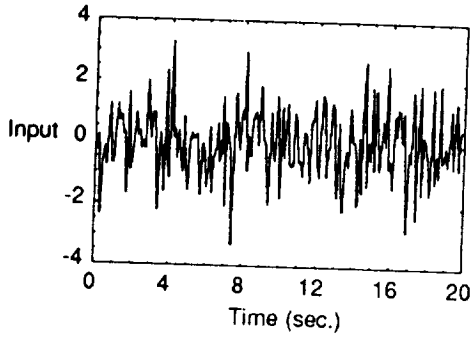

Figure 13: Excitation input time history.

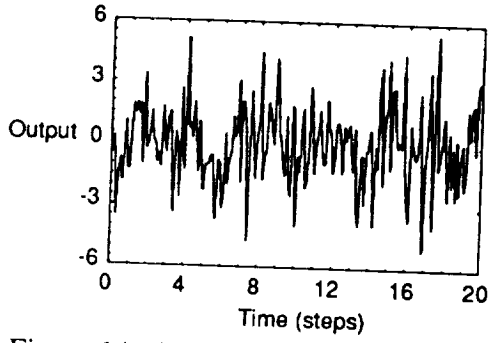

Figure 14: System response time history.

Using the above time histories, the network weights can be identified using Eq. (31). First, consider the case where p = 6, the following values for the network weights are obtained:

$$\hat{\alpha}_1 = 8.02\times10^{-1}, \ \hat{\alpha}_2 = -4.59\times10^{-3}, \ \hat{\alpha}_3 = -2.31\times10^{-2}$$
$$\hat{\alpha}_4 = 4.11\times10^{-1}, \ \hat{\alpha}_5 = 2.02\times10^{-1}, \ \hat{\alpha}_6 = -7.24\times10^{-1}$$

$$\hat{\beta}_0 = 1.50, \ \hat{\beta}_1 = -1.01, \ \hat{\beta}_2 = -5.48\times10^{-4}, \ \hat{\beta}_3 = 1.59\times10^{-1}$$
$$\hat{\beta}_4 = -5.02\times10^{-1}, \ \hat{\beta}_5 = -4.27\times10^{-1}, \ \hat{\beta}_6 = 8.75\times10^{-1} \tag{83}$$

The above results are checked against the data by performing an open-loop prediction of the response using the input alone,

$$\hat{y}(k) = \alpha_1\hat{y}(k-1) + \cdots + \alpha_6\hat{y}(k-6) + \beta_0 u(k) + \beta_1 u(k-1) + \cdots + \beta_p u(k-6) \tag{84}$$

and an one-step ahead prediction (or observer estimation) using both actual input and output data,

$$\hat{y}(k) = \alpha_1 y(k-1) + \cdots + \alpha_6 y(k-6) + \beta_0 u(k) + \beta_1 u(k-1) + \cdots + \beta_p u(k-6) \tag{85}$$

It can be verified that in both cases, both predicted responses match the actual data exactly. Again, it should be emphasized that the result shown in Eqs. (83) represents a set of weights that can be identified from any feedforward network that uses 6 past values of input and output data to predict the current response. Specifically, if one uses a network consisting of a single neuron, then the values listed in Eqs. (83) are precisely the weights of this neuron. On the other hand, if a feedforward network consisting of several layers of linear neurons is used to identify the system, then the values in Eqs. (83) are the weights of a single neuron representation that is mathematically equivalent to the multi-layer network.

The system in Eqs. (82) in fact contains one uncontrollable mode as revealed by the singular values of the controllability matrix, $\mathbf{C} = [A^5B, A^4B, \ldots, AB, B]$,

$$\sigma_1 = 1.08,\ \sigma_2 = 5.92\times10^{-1},\ \sigma_3 = 3.69\times10^{-1},\ \sigma_4 = 2.19\times10^{-1}$$
$$\sigma_5 = 9.44\times10^{-17},\ \sigma_6 = 1.91\times10^{-17} \tag{86}$$

The model in Eq. (84) is therefore an over-parameterized model. The same system can be modeled by using data from only 4 past time steps to predict the current response, i.e., p = 4. The corresponding weights are given below:

$$\hat{\alpha}_1 = 2.29,\ \hat{\alpha}_2 = -2.67,\ \hat{\alpha}_3 = 2.26,\ \hat{\alpha}_4 = -9.84\times10^{-1}$$
$$\hat{\beta}_0 = 1.50,\ \hat{\beta}_1 = -3.24,\ \hat{\beta}_2 = 3.72,\ \hat{\beta}_3 = -2.98,\ \hat{\beta}_4 = 1.19 \tag{87}$$

Note that the over-parameterization in Eq. (84) is in the form of having more distant past input and output data to predict the current response, corresponding to the case of a neuron having additional input channels. This is in contrast to the case where over-parameterization is in the form of having additional neurons added to the network.

8.2. Model Reference Adaptive Control of A Linear System. Next, we consider the application of the model reference adaptive control of the above system. The goal is to have the system track a desired trajectory prescribed via the reference model,

$$y_m(k+1) = 0.4y_m(k) + 0.5y_m(k-1) - 0.3y_m(k-2) + r(k+1) \tag{88}$$

where $r(k) = \sin(k/2\pi)$. First, consider the ideal case where disturbance and noises are not present. Since the system has a single output, the predictor network consists of only one linear neuron. In this example, 6 past input and output values are used to predict the current response. Recall that this is a case of over-parameterization since the effective order of the system is only 4. The system is assumed to be unknown to the controller at the beginning, and the weights are initially set to zero. Simultaneous prediction and control is carried out producing the results shown in Figs. 15a-d below. Figure 15a shows that the system response (dashed curve) quickly tracks the desired response (solid curve). The time histories of the prediction error and of the tracking error during the process are shown in Fig 15b and 15c, respectively. The control input time history is shown in Fig. 15d revealing that the adaptive mechanism quickly produces the necessary control input to make the system track the desired response.

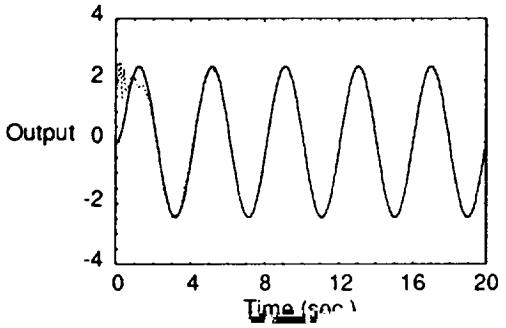

Figure 15a: Tracking response.

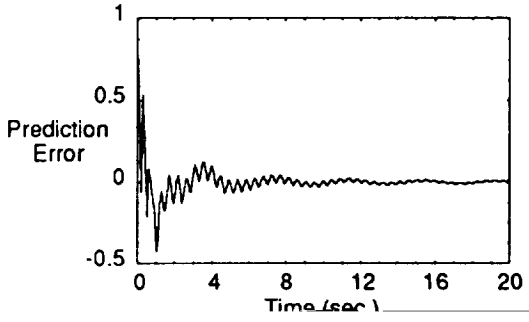

Figure 15b: Prediction error.

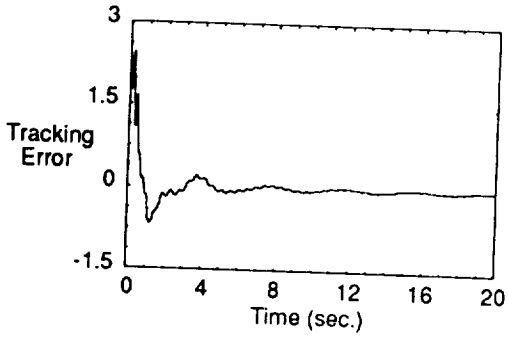

Figure 15c: Tracking error.

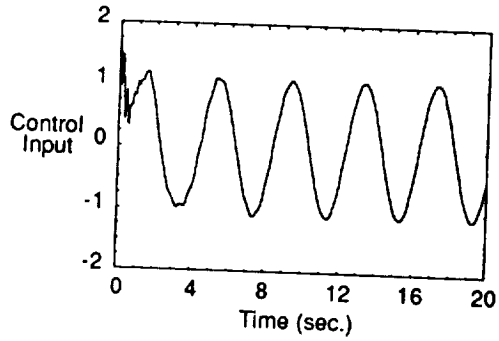

Figure 15b: Control input.

Figures 16 a-c show the adaptation when a disturbance, $d(k) = 0.5\cos(k/2\pi)$, and 5% measurement noise are added to the system. With the same adaptive controller, the system continues to track the desired response as shown in Fig. 16a. The effect of the noises can be seen in the random variation in the prediction error and the tracking error time histories, Figs. 16b and 16c. The new control input history that makes the system track the desired response and accommodate this disturbance is shown in Fig. 16d.

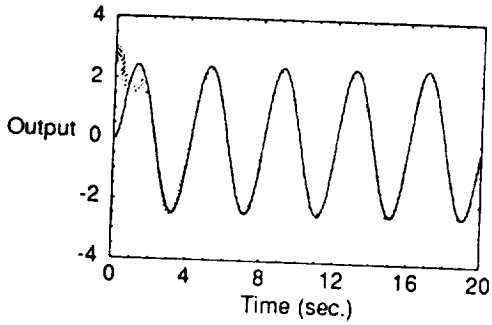

Figure 16a: Tracking response with disturbance and noise present.

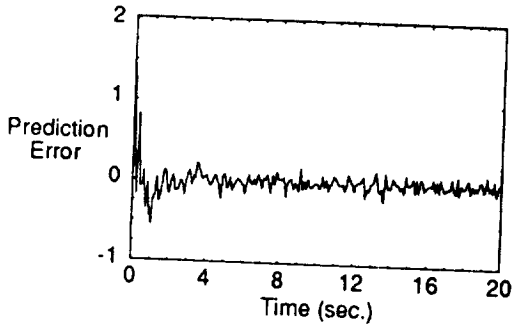

Figure 16b: Prediction error.

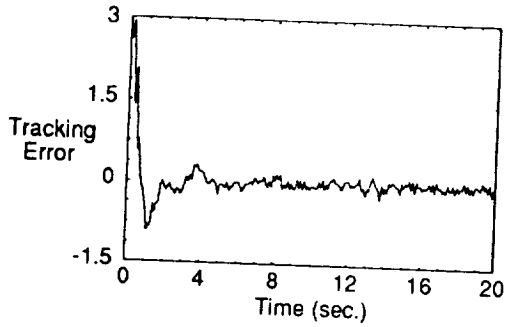

Figure 16c: Tracking error.

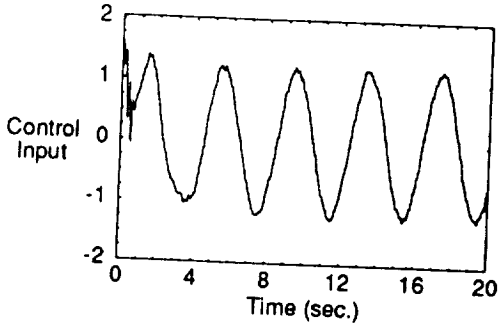

Figure 16b: Control input.

8.3. Identification and Prediction of a Non-Linear System. While it is not possible to have a linear model that can reproduce the open-loop response of a non-linear system, it is possible to have a linear predictor that can reasonably predict the non-linear response. The predictor model uses actual input and output data to compute the

linear system. Figures 18a-d show the tracking response, the prediction error, the tracking error, and the control input time histories, respectively. Recall that the control method does not require that the open-loop model be identified, but rather the predictor model that can reasonably predict the response, which is the case illustrated in the previous example.

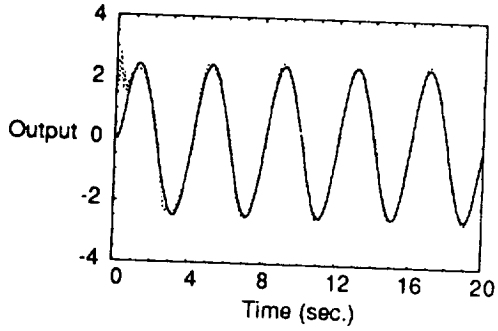

Figure 18a: Tracking response. (non-linear system)

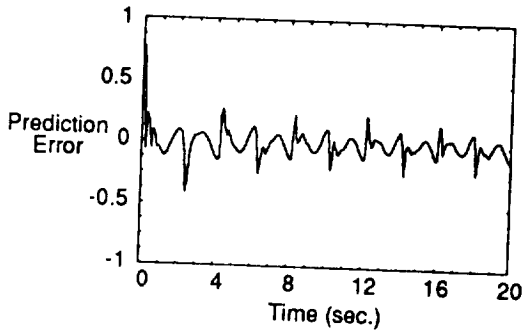

Figure 18b: Prediction error.

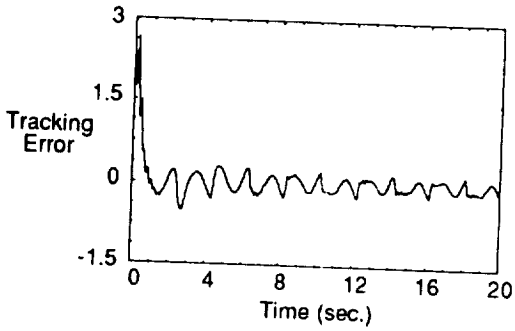

Figure 18c: Tracking error.

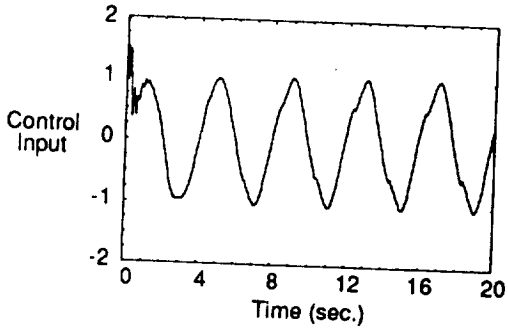

Figure 18b: Control input.

When disturbance and noise are added to the system, the resulting behavior of the system is shown in Figs. 19a-d. Again, this reveals a certain degree of stability robustness of the adaptive scheme to possible disturbance and noises. This is due to the inherent robustness in the ability of linear predictors that can predict the non-linear response.

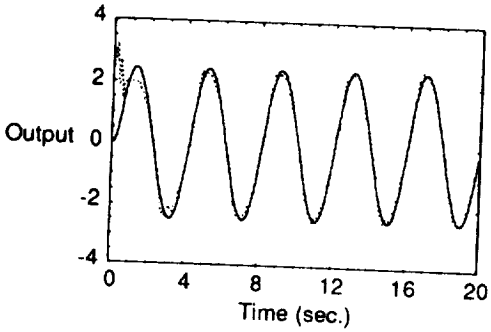

Figure 19a: Tracking response with dist. and noise present (non-linear syst.)

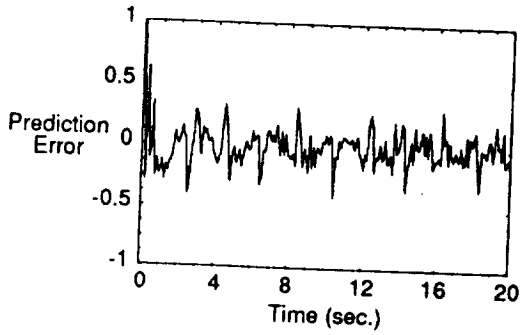

Figure 19b: Prediction error.

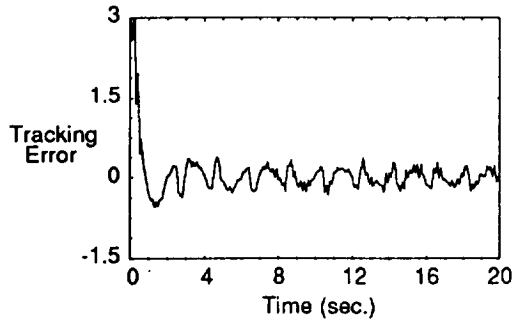

Figure 19c: Tracking error.

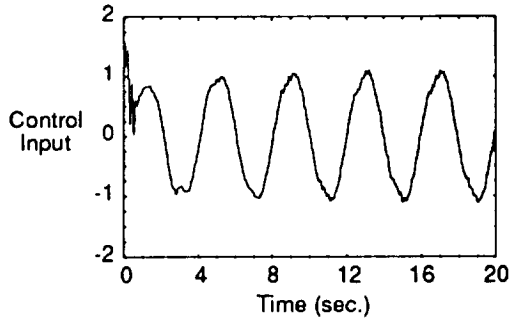

Figure 19b: Control input.

8. Summary and Concluding Remarks

This paper presents the basic concepts of the neural networks as related to the problem of modelling and control of a dynamic system. Two basic forms of the neural networks, the feedforward network and the recurrent network, are discussed. Emphasis is placed on the interpretation of the neural networks in terms of standard linear system theory so that better insight may be gained when these concepts are applied in practice. Relationship between the feedforward neural network and the state space model and between the recurrent network and the observer model is explained. To identify a linear system, the discussion in this paper reveals that it is neither advantageous nor necessary to use a multi-layer network, but rather a single layer of linear neurons is adequate. The resultant simplified network is then equivalent to standard regressive models that are often used in adaptive systems theory. The real benefit of a neural network in system identification is in its capacity to capture non-linearities, in which case the neurons must be non-linear. With respect to the control of both linear and non-linear systems, however, it is shown that it is not the identification of the open-loop system that governs the stability of the tracking behavior, but rather the ability of a mechanism that can predict future response based on actual available input-output data. It is shown that this mechanism can often be provided simply by a linear predictor. A linear predictor, consisting of a single layer of linear neurons, is in fact an optimal choice for a linear system. The same linear predictor can often be adequate for non-linear systems as well, making it directly applicable to the control a non-linear system. The resulting control technique is simply model reference adaptive control, a well-known technique in linear system control. Such a linear predictor can be easily determined from input-output data. If implemented on-line, the method can also adapt to changing dynamics. The recent advent of the Observer/Kalman filter identification (OKID) method has motivated the design of controllers that are based directly on the observer Markov parameters. This paper shows that one such design is model reference control because the observer Markov parameters are precisely the coefficients of an optimal linear predictor. The design takes advantage of the ability of the predictor to handle certain non-linear systems, an often observed fact in practical implementation of the OKID method.

To make adaptive control truly useful in practice, constraints with respect to sampling and computation speeds must be addressed. Naturally, the adaptive scheme places heavy emphasis on on-line measurements rather than some known model of the system for control. Sensor failure, therefore, becomes an important issue. Practical consideration may dictate a compromise between fixed-gain and adaptive control, thus requiring a mechanism to determine when adaptation should take place. The question of sensor placement and sensor selection are also important ones. This is to avoid the situation where the system can produce bounded output when driven by unbounded input.

This case requires additional theoretical treatment than that presented in this paper. Finally, the paper concerned mostly with stability rather than performance robustness issues. Further work is required to assess this aspect of the problem.

9. References

1 Narendra, K.S. and Parthasarathy, K., "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Transactions on Neural Network*, Vol. 1, No. 1, March 1990.

2 Hornik, K., Stinchcombe, M., and White, H., "Multilayer Feedforward Neural Networks Are Universal Approximators," *Neural Networks*, Vol. 2, No. 5, 1989.

3 Billings, S.A. and Leontaritis, I.J., "Input-Output Parametric Models for Non-Linear Systems. Part 1: Deterministic Non-Linear Systems; Part 2: Stochastic Non-Linear Systems, " *International Journal of Control*, Vol. 41, 1985.

4 Chen, S., Billings, and Grant, P.M., "Non-Linear System Identification Using Neural Network," *International Journal of Control*, Vol. 51, No. 6, 1990.

5 Hyland, D.C., "Neural Network Architectures for On-Line System Identification and Adaptively Optimized Control," *Proceedings of the IEEE Conference on Decision and Control*, Brighton, U.K., December 1991.

6 Goodwin, G.C. and Sin, K.S., *Adaptive Filtering, Prediction, and Control*, Prentice Hall, Englewood Cliffs, New Jersey, 1984.

7 Ljung, L. and Södertröm, T., *Theory and Practice of Recursive Identification*, The MIT Press, Cambridge, Massachusetts, 1983.

8 Chen, C.-W., Huang, J.-K., Phan, M. and Juang, J.-N., "Integrated System Identification and Modal State Estimation for Control of Large Flexible Space Structures," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 1, pp. 88-95, January-February 1992.

9 Juang, J.-N., Phan, M., Horta, L.G., and Longman, R.W., "Identification of Observer/Kalman Filter Markov Parameters: Theory and Experiments," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, New Orleans, Louisiana, August 1991; accepted for publication in the *Journal of Guidance, Control, and Dynamics*.

10 Phan, M., Horta, L.G., Juang, J.-N., and Longman, R.W., "Linear System Identification Via An Asymptotically Stable Observer," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, New Orleans, Louisiana, August 1991; also, accepted for publication in the *Journal of Optimization Theory and Applications*.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1993	Technical Memorandum

4. TITLE AND SUBTITLE
On Neural Networks in Indentification and Control of Dynamic Systems

5. FUNDING NUMBERS
WU 585-03-11-09

6. AUTHOR(S)
Minh Phan*, Jer-Nan Juang, David C. Hyland**

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23681-0001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA TM-107702

11. SUPPLEMENTARY NOTES
*Lockheed Engineering and Sciences Company, Hampton, Virginia.
**Harris Corporation, Melbourne, Florida.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited

Subject Category 39

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The paper presents a discussion on the applicability of neural networks in the identification and control of dynamic systems. Emphasis is placed on the understanding of how the neural networks handle linear systems and how the new approach is related to conventional system identification and control methods. Extensions of the approach to non-linear systems are then made. The paper explains the fundamental concepts of neural networks in their simplest terms. Among the topics discussed are feedforward and recurrent networks in relation to the standard state-space and observer models, linear and non-linear auto-regressive models, linear, predictors, one-step ahead control, and model reference adaptive control for linear and non-linear systems. Numerical examples are presented to illustrate the application of these important concepts.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Adaptive control; neural networks; system identification	33
	16. PRICE CODE A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	